أنظمة تمثيل الأعداد في الحاسب

و ملحقاتها من عمليات منطقية

و نظرة سريعة على وحدات التخزين و أجزائها

إعداد

إحياء

نسخة:1

1430-1431هـ

إحياء

فهرس

مقدمة:......3
1 أنظمة العد:......3
1.1 نظام العد العشري:......3
1.2 نظام العد الثنائي......3
1.3 نظام العد السداسي عشر:......4
1.4 نظام العد BCD......5
2 تنظيم البيانات 6
2.1 البت......7
2.2 nibbles......7
2.3 Word......7
2.4 double word......8
3 العمليات الحسابية على الأعداد الثنائية و السداسية عشر......8
3.1 العمليات المنطقية على البيتات :......8
3.1.1 AND......8
3.1.2 OR......9
3.1.3 XOR......9
3.1.4 NOT......9
3.2 العمليات المنطقية على الأعداد الثنائية......9
4 الأعداد ذات الإشارة و الأعداد دون الإشارة signed and unsigned numbers......10
5 البيانات المحزومة......11
ملحق(أ)......11
ملحق(ب)......12

مقدمة:

هذا الموضوع من المواضيع المهمة التي ينبغي على المتعلم لعلوم الحاسوب أو بعض لغات البرمجة ذات المستوى الأدنى تعلمها مثل الأسمبلي .[1]
و يحتوي هذا المقال على : أنظمة العد الثنائية و السداسي عشر ، تنظيم البيانات الثنائية(bits,nibbles,bytes,words,double words)، أنظمة العد ذات الإشارة و دون الإشارة[2] ، الحساب ، المنطقية ، تحزيم البيانات.

1 أنظمة العد:

معظم أنظمة الحاسوب الحديثة لا تستخدم النظام العشري(الذي يتألف من 0 ←9) في تمثيل البيانات و لكن بدلاً منها فإنها تستخدم النظام الثنائي(الذي يتألف من 0 و 1).

1.1 نظام العد العشري:

كما قلنا أن يتألف من 10 أرقام 0 ←9 في تمثيل البيانات إضافة إلى أنها تستخدم 10 أساساً لها فعندما نتكلم عن عدد مثل "123" فإنه

$$1*10^0+2*10^1+3*10^2$$

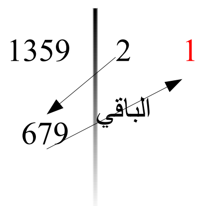
1.2 نظام العد الثنائي

إن الحواسيب الجديدة(مثلة حواسيب IBM) تستخدم المنطق الثنائي في تمثيل القيم متمثلة في 0 فولت أو +5 فولت و بالتالي فإنه يمكننا بالتالي التعبير من خلالها عن قيمتين مختلفتين و جرت الامور أن تكون 0 و 1 و هذا مصادفة وافق نظام العد الثنائي و بما أن نظام العد و منطق المعالج 80X86 متوافقان فكان من الطبيعي أن تستخدم حواسيب IBM نظام العد الثنائي في تمثيل الأرقام ، وللتحويل من النظام العشري إلى النظام الثنائي نتبع الطريقة الواردة في المثال التالي:

ليكن العدد لدينا 1359 و نريد تحويله إلى النظام الثنائي
فإننا نقوم باستخدام باقي القسمة الصحيح لإيجاد العدد الثنائي المكافئ فلو كان هناك باقي فإننا نسند 1 عند تلك المرحلة و نتابع القسمة على الناتج الصحيح و إن لم يكن هناك باقي أسندنا 0 في تلك المرحلة و نتابع القسمة على الناتج الصحيح أيضاً و بالنهاية عند الوصول للصفر نأخذ هذه الأرقام 0 -1 و نضعها و نستطيع بعد ذلك من التأكدمن أن الناتج صحيح باستخدام التحويل من النظام الثنائي إلى العشري و ذلك كما يلي:

1 لقد تم تقسيم لغات البرمجة إلى صنفين : **عالية المستوى** و **منخفضة المستوى** و ذلك حسب قربها من اللغة المحكية ، فما كان منها قريب من لغة البشر كان عالي المستوى مثل: السي و الجافا و غيرها و الصنف الآخر سمي بعالي المستوى مثل الأسمبلي و لغة الآلة

2 ذات الإشارة او دونها(الموجب و السالب)



1359	2	1
679	2	1
339	2	1
169	2	1
84	2	0
42	2	0
21	2	1
10	2	0
5	2	1
2	2	0
1	2	1
0		

الناتج=10101001111

$$10101001111_{2}=1359_{10}$$

للتأكد(التحويل من الثنائي إلى العشري):
و بطريقة عكسية نحول من الثنائي إلى العشري

$$1\times2^{10}+0\times2^{9}+1\times2^{8}+0\times2^{7}+1\times2^{6}+0\times2^{5}+0\times2^{4}+1\times2^{3}+1\times2^{2}+1\times2^{1}+1\times2^{0}$$
$$1024+0+256+0+64+0+0+8+4+2+1=1359_{10}$$

1.3 نظام العد السداسي عشر:

يتألف هذا النظام من 16 عنصر للتمثيل و أساسه أيضاً 16 و هذه العناصر هي :0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F
و تأتي فائدة هذا النظام في حل مشكلة التعامل مع الأعاد الكبيرة جداً فكما نعرف أن القيم يتعامل معها الحاسوب على شكل 0 و 1 و لذلك فقد تصبح هذه القيمة الكبيرة واسعة جداً عند تمثيلها بالثنائي ، لذلك فكر مهندسو الحاسوب أن يتم تحويلها إلى النظام العشري و لكنهم وجدوا أن النظام السداسي عشر أفضل من حيث أنه يمثل القيم بشكل محزوم أكثر و هذا طبيعي لأنه لديه 16 عنصر لتمثيل الأعداد مقابل النظام العشري الذي يحوي 10 عناصر فقط بالإضافة إلى سهولة التحويل ، و لكن وجب في هذه الحالة أن نميز الأعداد الممثلة بالعشري عن السداسي عشر عن الثنائي في الحاسوب ؛ فما يدريك هل 10 بالعشري أم بالسداسي عشر أم بالعشري لذلك تم الإصطلاح على وضع h بعد الأعداد الممثلة بالسداسي عشر hexadecimal و b بعد الممثلة بالثنائي binary و d بعد الممثل بالعشري decimal .

12Fh , 123d , 0110b

و من هذا الجدول يتضح لك سهولة التحويل

النظام السداسي عشر	المقابل بالعشري	المقابل بالثنائي
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

و لنأخذ الآن مثال لتتبين لنا سهولة التحويل : ليكن لدينا العدد AF31 و المطاوب تحويله إلى الثنائي

A F 3 1

1010 1111 0011 0001

و كذلك هو التحويل بالعكس .

1.4 نظام العد BCD

و هو اختصار binary coded decimal و هذا النظام مستخدم من أجل سهولة التحويل من الطرفين ، بعكس ما رأينا سابقاً .

لنأخذ العدد 127 و لنحاول تحويله إلى نظام العدد الثنائي و نظام العد BCD و لنجد الفرق :

إلى BCD

1 2 7

0001 0010 0111

إلى الثنائي

127	2	1
63	2	1
31	2	1
15	2	1
7	2	1
3	2	1
1	2	1

0111 1111

فانظر إلى الاختلاف في الناتجين و انظر أيضاً إلى سهولة التحويل فإن التحويل إلى BCD يعتمد على الجدول التالي الذي يضع القيمة الثنائية لكل عدد عشري و لذلك فإننا نأخذ كل أربع خانات ثنائية و نمثلها بعدد عشر يبناءً على نظام BCD

BCD	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

2 تنظيم البيانات

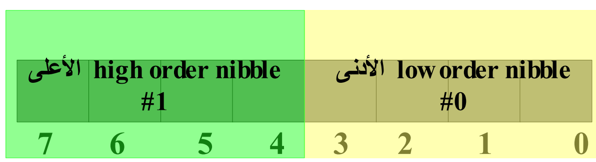
يستخدم في شرائح معالج انتل تقسيمات للبيتات (البايت و النيبل و الوورد و الدوبل وورد)

2.1 البت

يعتبر البت أصغر وحدة تخزينية في الحاسوب و يتألف من احتمالين منطقيين جرى التعارف عليها كما قلنا على أنه 0 و 1 و لكن يمكن أن تكون أي شيء آخر مثل أحمر و أخضر أو 8 و 9 أو أحمر و 0 أو أي شيء آخر و هي في النهاية فيزيائياً تتألف من احتمالين فولتيين (v0 أو v5) .

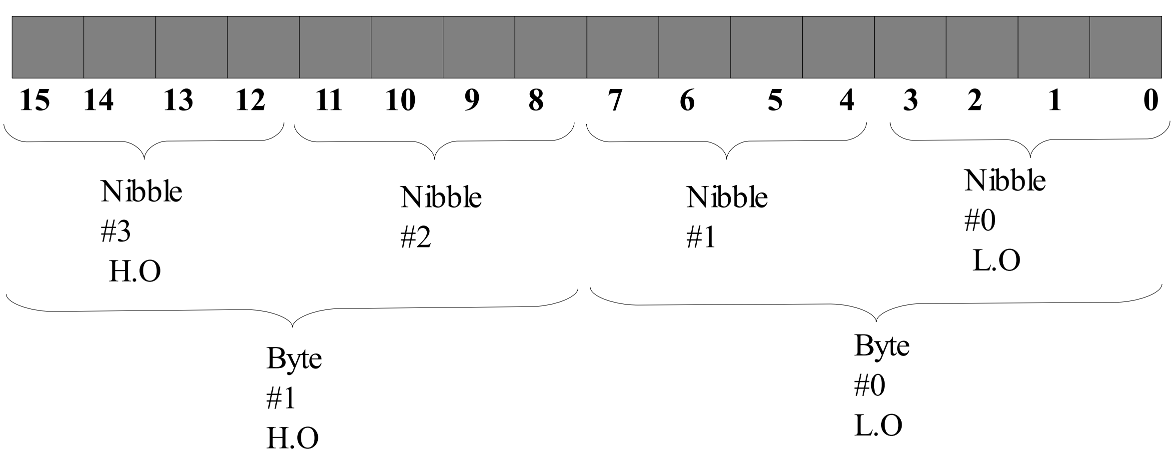
nibbles 2.2

إنها تتألف من مجموع 4 بيتات سويةً و تستخدم هذه البنية في تمثيل عدد واحد من أعداد النظام السداسي عشر أو BCD (binary coded decimal) و بالتالي فإن البايت يتألف من اثنين من nibble و يقسم لأعلى و أدنى كما في الشكل التالي و يبدأ الترقيم من الـصفر .



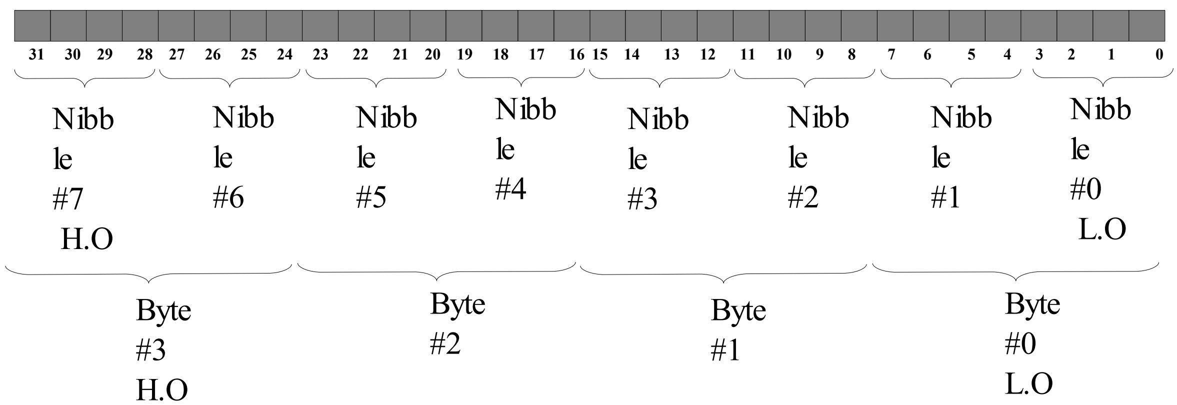
Word 2.3

تتألف من 16 بيت سوية و بالتالي من بايتين و أربع nibble



double word 2.4

تتألف من 32 بيت موزعة كما يلي:



3 العمليات الحسابية على الأعداد الثنائية و السداسية عشر

هناك العديد من العمليات التي يمكن أن تنفذها على الأعداد الثنائية أو السداسية عشر مثل الضرب و الجمع و الطرح و التقسيم و من المفضل طبعاً أن تمتلك آلة حاسبة تقوم بهذا النوع من الحسابات لأنها معقدة و طويلة جداً .

3.1 العمليات المنطقية على البيتات :

هناك أربع عمليات رئيسية منطقية على البيتات نقوم بها و هي :AND,OR,XOR(EXCLUSIVE-OR),NOT
و هذا بيان بكل منها :

AND 3.1.1

كما نعلم أن لكل بايت احتمالين 0 أو 1 و بالتالي عملية AND التي تتعلق ببايتين له عدة احتمالات جُمعت في هذا الجدول ولا تتحقق إلا عندما يكون الطرفين يساوي واحد

AND	1	0
1	1	0
0	0	0

3.1.2 OR

تتحقق أو يكون الناتج واحد عنما يكون أحد الطرفين واحد على الأقل(أي ممكن الطرفين معا واحد أو طرف واحد فقط واحد)

OR	1	0
1	1	0
0	0	0

3.1.3 XOR

لا تتحقق أو يصبح الناتج واحد إلا عندما يكون أحد الطرفين <u>فقط</u> واحد .

XOR	1	0
1	0	1
0	1	0

3.1.4 NOT

NOT 1=0

NOT 0=1

و هو النفي .

3.2 العمليات المنطقية على الأعداد الثنائية

عندما تحدثنا في الفقرة 3.1 عن العمليات المنطقية تكلمنا عن عملية منطقية بين بيتين و لكن ماذا لو كان نتحدث عن سلسلة من البيتات (8,16,32) حسب طول البيانات التي يعالجها المعالج ، لذلك فإن المعالج 80X86 يعالج كل بيتين سوية لسلسلة من البيتات و هذا مثال على ذلك :

```
 0101 1011
 1110 1110
-------------
 0100 1010
```

4 الأعداد ذات الإشارة و الأعداد دون الإشارة signed and unsigned numbers

في الفقرات السابقة جميعها لم نأت على ذكر الأعداد السالبة فكل ما تعاملنا معه كان قيم موجبة فقط و لكن ماذا لو أردنا التحدث عن القيم السالبة ، لقد فكر علماء الحاسوب بهذا الأمر و وجدوا أنه من العدل جعل 256 حالة التي يمكن لبايت أن يمثلها جعل نصفها للأعداد السالبة و النصف الآخر للموجبة أي -128 ←-1 و من 0 ←127 و هكذا الحال لو كان 16 بيت أو 32 بيت .

حسناً كيف نميز العدد الموجب من السالب و الممثل بالنظام الستة عشري أو بالنظام الثنائي ، القاعدة تقول أنه إذا كان البيت الأعلى يساوي 0 فهو موجب و إذا كان يساوي واحد فهو سالب ، أمثلة:

```
  8     0     0     0    h
1000  0000  0000  0000

   1     0     0    h
0001  0000  0000

  7     F     F     F    h
0111  1111  1111  1111
```

و بالتالي العدد الأول سالب و البقية موجبين.

و آلية تحويل العدد من الموجب إلا السالب تكون كما يلي:

1-تطبيق العملية المنطقية NOT على جميع البيتات 2-إضافة بيت واحد كما يلي:

لتحويل -5 إلى 5 :

```
 0000 0101  -5
 1111 1010
-------------
 1111 1011  +5
```

5 البيانات المحزومة

من الصحيح أننا وحدات التخزين قد قُسمت إلى bit,byte,nibble,word,double word و لكن إحيانا قد ينتج هدر في مساحة الذاكرة و مثالنا : لو أردنا كتابة تاريخ اليوم مثلاً : 16/9/10 و بالتالي لدينا ثلاثة قيم عددية تحتاج إلى 3 بايتات و لكن لنفكر سويةً هل الأشهر تحتاج إلى 256 احتمال أو الأشهر كذلك أو السنوات كذلك على اعتبار أنها من 00 إلى 99 لذلك فإن الأيام تحتاج 1-7 احتمال و الاشهر 1-12 احتمال و الأعوام 00-99 احتمال و بالتالي :
4 بيتات للأيام – 5 بيتات للأشهر -7 بيتات للسنوات و بالتالي كلهم في وورد ،كما يلي :

D	D	D	D	M	M	M	M	M	Y	Y	Y	Y	Y	Y	Y
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

و لكن كما تروا أن الامر يحتاج إلى فك هذا التحزيم و فهمه و كله هذا يحتاج إلى وقت و إمكانيات.

ملحق(أ)

جدول بالمصطلحات الأجنبية

العربية	الأجنبية
تمثيل البيانات	Data representation
أسمبلي	assembly
لغة الآلة	Machine language
العشري	decimal
الثنائي	binary
السداسي عشر	hexadecimal
العمليات المنطقية	Logical operation
العمليات الحسابية	Arithmetic operation
البيانات المحزومة	Packed data

ملحق(ب)

البحث معظمه مبني على الفصل الاول من كتاب **art of assembly language** من خلال الترجمة مع العلم أن الرسوميات كلها تابعة لإنتاجية الفريق.

و هذا الملف طبعاً خاضع للتطوير كل فترة ، و نذكّر بما يلي :

مسؤولية الفريق

الفريق لا يتحمل أي تبعة من تبعات ورود أخطاء لأن الفريق في طور النشأة و كل ابن آدم خطّاء،و لا ينصح باستخدام إنتاجياته مصادر تعليمية

في حال ورود خطأ

يرجى التبليغ على بريد الفرق e7aaproj@gmail.com و لكم جزيل الشكر و كذلك لمن أراد الانضمام .

تحديثات

سيتم بإذنه تعالى تحديث الكتاب كل فترة لذا يرجى الانتباه لهذا النقطة.

توصية

سبحانك اللهم و بحمدك سبحان ربي العظيم أستغفرك و أتوب إليك . اللهم لا علم لنا إلا ما علمتنا .

احرص أخي المسلم ألا تستخدم علمك إلا فيما يرضي الله و يعز المسلمين و يهزم أعداءهم و هذا كل ما نرجوه منك بعد الدعاء لمن ساهم في هذا العمل بالتوفيق و الرشاد و الفوز بالجنة و بمغفرة قيوم السماوات و الأرض.

إحياء